Analyze A/B Test Results

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question. The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the RUBRIC.

Part I - Probability

To get started, let's import our libraries.

In [1]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
#We are setting the seed to assure you get the same answers on quizzes as we set up
random.seed(42)
```

`1.` Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset and take a look at the top few rows here:

In [2]:
```
df=pd.read_csv('ab_data.csv')
df.head(10)
```

Out[2]:

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1
5	936923	2017-01-10 15:20:49.083499	control	old_page	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
7	719014	2017-01-17 01:48:29.539573	control	old_page	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1

b. Use the below cell to find the number of rows in the dataset.

In [3]:
```
len(df)
```

Out[3]: `294478`

c. The number of unique users in the dataset.

In [4]:
```
len(pd.unique(df.user_id))
```

Out[4]: `290584`

d. The proportion of users converted.

In [5]:
```
df['converted'].value_counts(normalize=True)
```

Out[5]:
```
0    0.880341
1    0.119659
Name: converted, dtype: float64
```

e. The number of times the `new_page` and `treatment` don't line up.

In [6]:
```
df.groupby(['group','landing_page']).size()
```

Out[6]:
```
group      landing_page
control    new_page          1928
           old_page        145274
treatment  new_page        145311
           old_page          1965
dtype: int64
```

f. Do any of the rows have missing values?

In [7]:
```
df.isnull().sum(axis = 0)
```

Out[7]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

`2.` For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to provide how we should handle these rows.

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

In [8]:
```
df2 = df[(df["group"]=="treatment") & (df["landing_page"]=='new_page') | \
                ((df.group == 'control') & (df.landing_page == 'old_page'))]


df2.shape
```

Out[8]: `(290585, 5)`

In [9]:
```
# Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].s
```

Out[9]: `0`

`3.` Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

a. How many unique **user_id**s are in **df2**?

In [10]:
```
df2.user_id.nunique()
```

Out[10]: `290584`

b. There is one **user_id** repeated in **df2**. What is it?

In [11]:
```
dupes = df2['user_id'].duplicated()
df2['user_id'][dupes]
```

Out[11]:
```
2893    773192
Name: user_id, dtype: int64
```

c. What is the row information for the repeat **user_id**?

In [12]:
```
df2.loc[df['user_id'] == 773192]
```

Out[12]:

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

d. Remove **one** of the rows with a duplicate **user_id**, but keep your dataframe as **df2**.

In [13]:
```
df2.drop_duplicates()
```

Out[13]:

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1
...	...	...	...	...	...
294473	751197	2017-01-03 22:28:38.630509	control	old_page	0
294474	945152	2017-01-12 00:51:57.078372	control	old_page	0
294475	734608	2017-01-22 11:45:03.439544	control	old_page	0
294476	697314	2017-01-15 01:20:28.957438	control	old_page	0
294477	715931	2017-01-16 12:40:24.467417	treatment	new_page	0

290585 rows × 5 columns

`4.` Use **df2** in the below cells to answer the quiz questions related to **Quiz 4** in the classroom.

a. What is the probability of an individual converting regardless of the page they receive?

In [14]:
```
df2['converted'].mean()
```

Out[14]: `0.11959667567149027`

b. Given that an individual was in the `control` group, what is the probability they converted?

In [15]:
```
len(df2[(df2['group']=='control') & (df2['converted']==1)])/ len(df2[df2['group']=='co
```

Out[15]: `0.1203863045004612`

c. Given that an individual was in the `treatment` group, what is the probability they converted?

In [16]:
```
len(df2[(df2['group']=='treatment') & (df2['converted']==1)])/ len(df2[df2['group']==
```

Out[16]: `0.11880724790277405`

d. What is the probability that an individual received the new page?

In [17]:
```
len(df2[df2['landing_page']=='new_page'])/len(df2)
```

Out[17]: `0.5000636646764286`

e. Consider your results from a. through d. above, and explain below whether you think there is sufficient evidence to say that the new treatment page leads to more conversions.

12% that received the old_page were converted. 11% that received the new_page were converted. so no ufficient evidence to say that the new treatment page leads to more conversions.

Part II - A/B Test

Notice that because of the time stamp associated with each event, you could technically run a hypothesis test continuously as each observation was observed.

However, then the hard question is do you stop as soon as one page is considered significantly better than another or does it need to happen consistently for a certain amount of time? How long do you run to render a decision that neither page is better than another?

These questions are the difficult parts associated with A/B tests in general.

`1.` For now, consider you need to make the decision just based on all the data provided. If you want to assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%, what should your null and alternative hypotheses be? You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the converted rates for the old and new pages.

Null hypothesis: the old page is better or as good as the new page

Alternative hypothesis: new page is better that the old page

`2.` Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab_data.csv**.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. You can use **Quiz 5** in the classroom to make sure you are on the right track.

a. What is the **convert rate** for p_{new} under the null?

In [18]:
```
p_new = df2['converted'].mean()
p_new
```

Out[18]: `0.11959667567149027`

b. What is the **convert rate** for p_{old} under the null?

In [19]:
```
p_old = df2['converted'].mean()
p_old
```

Out[19]: `0.11959667567149027`

c. What is n_{new}?

In [20]:
```
n_new = df2[df2['landing_page'] == 'new_page'].shape[0]
n_new
```

Out[20]: `145311`

d. What is n_{old}?

In [21]:
```
n_old = df2[df2['landing_page'] == 'old_page'].shape[0]
n_old
```

Out[21]: `145274`

e. Simulate n_{new} transactions with a convert rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

In [22]:
```
new_page_converted = np.random.binomial(1,p_new,n_new)
new_page_converted.mean()
```

Out[22]: `0.11974317154241593`

f. Simulate n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

In [23]:
```
old_page_converted = np.random.binomial(1,p_old,n_old)
old_page_converted.mean()
```

Out[23]: `0.12158404119112849`

g. Find p_{new} - p_{old} for your simulated values from part (e) and (f).

In [24]:
```
new_page_converted.mean() - old_page_converted.mean()
```

Out[24]: `-0.0018408696487125648`

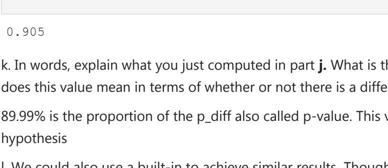
h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

In [25]:
```
p_diffs = []

for _ in range(10000):
    new_page_converted = np.random.binomial(1,p_new,n_new).mean()
    old_page_converted = np.random.binomial(1,p_old,n_old).mean()
    p_diffs.append(new_page_converted - old_page_converted)
```

i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

In [26]:
```
p_diffs = np.array(p_diffs)
plt.hist(p_diffs)
plt.xlabel('p_diffs')
plt.ylabel('Frequency')
plt.title('Simulated Difference of new_page & old_page converted under the Null');
```



j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

In [27]:
```
df_control = df2.query('group == "control"')
df_treatment = df2.query('group == "treatment"')

obs_diff = df_treatment.converted.mean() - df_control.converted.mean()
obs_diff
```

Out[27]: `-0.0015790565976871451`

In [28]:
```
(p_diffs > obs_diff).mean()
```

Out[28]: `0.905`

k. In words, explain what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

89.99% is the proportion of the p_diff also called p-value. This value means that we cann't reject the null hypothesis

l. We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance. Fill in the below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

In [29]:
```
import statsmodels.api as sm

convert_old = df2.query('group == "control" & converted == 1')['converted'].count()
convert_new = df2.query('group == "treatment" & converted == 1')['converted'].count()

n_new = len(df2.query('landing_page == "new_page"'))
n_old = len(df2.query('landing_page == "old_page"'))
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here is a helpful link on using the built in.

In [30]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_ol
z_score, p_value
```

Out[30]: `(-1.3116075339133115, 0.905173705140591)`

n. What do the z-score and p-value you computed in the previous question mean for the conversion rates of the old and new pages? Do they agree with the findings in parts **j.** and **k.**?

they don't reject the Null. The Null is that the converted rate of the old_page is is better or as good as that of the new. The p_value is 0.9 and is higher than 0.05 significance level, we cann't be confident with a 95% confidence level

Part III - A regression approach

`1.` In this final part, you will see that the result you acheived in the previous A/B test can also be acheived by performing regression.

a. Since each row is either a conversion or no conversion, what type of regression should you be performing in this case?

The dependent variable is binary, so you need to use logistic regression.

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

In [31]:
```
df2['intercept'] = 1
df2[['ab_page', 'old_page']] = pd.get_dummies(df2['landing_page'])
df2.head()
```

```
<ipython-input-31-3d232eb1f5cb>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  df2['intercept'] = 1
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:3191: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
r_guide/indexing.html#returning-a-view-versus-a-copy
  self[k1] = value[k2]
```

Out[31]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page	old_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0	1
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0	1
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0	1

c. Use **statsmodels** to import your regression model. Instantiate the model, and fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

In [32]:
```
log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = log_mod.fit()
results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
```

Out[32]:

Model:	Logit	Pseudo R-squared:	0.000
Dependent Variable:	converted	AIC:	212780.6032
Date:	2021-10-21 15:32	BIC:	212801.7625
No. Observations:	290585	Log-Likelihood:	-1.0639e+05
Df Model:	1	LL-Null:	-1.0639e+05
Df Residuals:	290583	LLR p-value:	0.18965
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-1.9888	0.0081	-246.6690	0.0000	-2.0046	-1.9730
ab_page	-0.0150	0.0114	-1.3116	0.1897	-0.0374	0.0074

In [33]:
```
np.exp(0.0150)
```

Out[33]: `1.015113064615719`

d. Provide the summary of your model below, and use it as necessary to answer the following questions.

Summary: Holding all other variables constant, the number of converted is 1.015 times more likely to be converted than those that are not converted. This means that the old page and new page are both equal in chance of converting users. We should not assume that the new page is better than the old page.

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

Hint: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in the **Part II**?

p-value = 0.189 In part II the p-value is 0.91. This might be because the tests of the regression model (not the A/B test) assumes an intercept and because of differences in one or two-tailed testing. in both values , alternative hypotheses is not chosen

f. Now, you are considering other things that might influence whether or not an individual converts. Discuss why it is a good idea to consider other factors to add into your regression model. Are there any disadvantages to adding additional terms into your regression model?

it's better, as it get more realistic disadvantage: more complex model

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy variables.** Provide the statistical output as well as a written response to answer this question.

In [34]:
```
countries_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
```

In [35]:
```
### Create the necessary dummy variables
df_new[['CA','UK', 'US']] = pd.get_dummies(df_new['country'])
# drop the country column since this is not necessary
df_new = df_new.drop('country', 1)
df_new.head()
```

Out[35]:

user_id	timestamp	group	landing_page	converted	intercept	ab_page	old_page	CA	UK	US
630000	2017-01-19 06:26:06.548941	treatment	new_page	0	1	1	0	0	0	1
630001	2017-01-16 03:16:42.560309	treatment	new_page	1	1	1	0	0	0	1
630002	2017-01-19 19:20:56.438330	control	old_page	0	1	0	1	0	0	1
630003	2017-01-12 10:09:31.510471	treatment	new_page	0	1	1	0	0	0	1
630004	2017-01-18 20:23:58.824994	treatment	new_page	0	1	1	0	0	0	1

h. Though you have now looked at the individual factors of country and page on conversion, we would now like to look at an interaction between page and country to see if there significant effects on conversion. Create the necessary additional columns, and fit the new model.

Provide the summary results, and your conclusions based on the results.

In [37]:
```
###df_new['intercept'] = 1

df_new['ab_UK'] = df_new['ab_page'] * df_new['UK']
df_new['ab_US'] = df_new['ab_page'] * df_new['US']
lm3 = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page', 'UK' , 'US', 'ab_U
results = lm3.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366108
         Iterations 6
```

Out[37]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290585
Model:	Logit	**Df Residuals:**	290579
Method:	MLE	**Df Model:**	5
Date:	Thu, 21 Oct 2021	**Pseudo R-squ.:**	3.483e-05
Time:	15:34:46	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.1918

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.0040	0.036	-55.008	0.000	-2.075	-1.933
ab_page	-0.0674	0.052	-1.297	0.195	-0.169	0.034
UK	0.0118	0.040	0.296	0.767	-0.066	0.090
US	0.0175	0.038	0.465	0.642	-0.056	0.091
ab_UK	0.0783	0.057	1.378	0.168	-0.033	0.190
ab_US	0.0469	0.054	0.872	0.383	-0.059	0.152

Conclusions

we used 3 methods to know if the new page is better or not. each of them provided us with the same asnwer. that the new page is not better and provides less conversion rates and we should stick to the old one

In [ ]: